FIRST STEPS TOWARDS A NEW FINITE ELEMENT SOLVER FOR MOEVE PIC TRACKING*

Ursula van Rienen[1]†, Dawei Zheng, Johann Heller, Christian Bahls
Institute of General Electrical Engineering, University of Rostock, 18059 Rostock, Germany
[1]also at Department Life, Light & Matter, University of Rostock, 18051 Rostock, Germany

Abstract

A relevant task in designing high-brilliance light sources based on high-current linear accelerators (e.g. Energy Recovery Linacs (ERLs)) consists in systematic investigations of ion dynamics in the vacuum chamber of such machines. This is of high importance since the parasitic ions generated by the electron beam turned out to be a current-limiting factor for many synchrotron radiation sources. In particular, the planned high current operation at ERL facilities requires a precise analysis and an accurate development of appropriate measures for the suppression of ion-induced beam instabilities. The longitudinal transport of ions through the whole accelerator plays a key role for the establishment of the ion concentration in the machine. Using the Particle-in-Cell (PIC) method, we started redesigning our code MOEVE PIC Tracking in order to allow for the fast estimation of the effects of ions on the beam dynamics. For that, we exchanged the previously used Finite Difference (FD) method for the solution of Poisson's equation within the PIC solver by a solver based on the Finite Element Method (FEM). Employing higher order FEM, we expect to gain improved convergence rates and thus lower computational times. We chose the Open Source Framework FEniCS for our new implementation.

INTRODUCTION

MOEVE was developed as a Particle-in-Cell (PIC) solver at the University Rostock. It is an abbreviation and stands for **M**ultigrid for n**o**n-**e**quidistant grids to sol**v**e Poisson's **e**quation. The software was originally developed in C by G. Pöplau et al. [1] and employs the Finite-Difference Technique (FD) to numerically discretize Poisson's equation. The discretized system of linear equations is solved iteratively by a geometric multigrid method.

MOEVE has been used successfully to simulate the interaction of electron beams with ionized residual gas [2, 3], several investigations for the clearing of ions with clearing electrodes and/or clearing gaps [4] and the simulation of transverse wake functions [5]. MOEVE has also been implemented in the tracking code GPT [6] and ASTRA [7].

Ion Clearing

Any residual gas in the vacuum chamber of an accelerator can be ionized rapidly by the electron beam. The resulting ion distribution is denoted as ion cloud. For many synchrotron radiation sources, these parasitic ions generated by the electron beam are a current-limiting factor. They often lead to beam instabilities, beam loss and they prevent a continuous filling of electron bunches into the ring shape machine.

In the existing synchrotron accelerators, mainly two strategies are used to ensure a minimum stability in standard operational regimes: clearing gaps and special electrodes for removing and neutralizing the ions. In certain high-current operating conditions ion effects are important, as they lead to beam instabilities. In particular, the planned high-current operation at ERL facilities requires a precise analysis and an accurate development of appropriate measures for the suppression of ion-induced beam instabilities [8].

The longitudinal transport of ions through the whole accelerator plays a key role for the establishment of the ion concentration in the machine. This aspect of the dynamics has implications on both the beam dynamics and the ion clearing efficiency but it has not been deeply studied up to now. In particular, the extent to which the accelerating resonators contribute to the transport is largely unclear. Thus, we are targeting a fast, systematic investigation of ion dynamics in the vacuum chamber of the machines involving the impact on the beam and and its application to reduce the effects related to ionized residual gas in high-current electron machines. This study follows our previous investigations on ion trapping in high-current storage rings and linear accelerators [2–4, 9, 10].

Prior Limitations of MOEVE Due to Finite Differences

Any PIC software consists of five different main modules. These are

1. Charge weighting
2. Discretization of Poisson's equation
3. Solver for Poisson's equation
4. Field interpolation at particles position
5. Update scheme of the particle distribution

MOEVE's current limitations, caused by the underlying FD discretization, affect the discretization and solution of Poisson's equation. A comparably large number of degrees of freedom (DOFs) are required for the accurate solution, especially because the tensor product grid in the FD method

* Work supported by the German Federal Ministry for Research and Education BMBF under contract 015K16HRA.

† ursula.van-rienen@uni-rostock.de

does not approximate the geometry well. The current FD implementation in MOEVE does not allow for the discretization of arbitrary domains without strong reductions in accuracy [11], as it would be required for the tasks at hand.

The PIC method scales in its complexity with the number of macro-particles and the required mesh cells of the discretization. While the number of macro-particles can not be further reduced (since this would lead to an insufficient accuracy) one can reduce the number of mesh cells by using a different discretization technique (e.g. the Finite Element Method (FEM)).

Using appropriate ansatz functions in FEM, e.g. *Crouzeix-Raviart* elements as in [12], one could even improve this convergence by at least one order[1] compared to the FD discretization, thus allowing a reduction in the number of mesh cells.

REPLACING THE FD SOLVER WITH FEM FROM FENICS

When using a PIC method, we compute the accelerating field of a charge density $\rho(x)$ from the solution of Poisson's equation on the domain Ω:

$$-\Delta u(x) = \frac{\rho(x)}{\varepsilon} \quad \forall x \in \Omega. \tag{1}$$

Usually, in our applications ε will be the vacuum permittivity ε_0.

For a unique solution $u(x)$ we have to impose boundary conditions on $\partial\Omega$:

$$u(x) = g_D(x) \quad \forall x \in \partial\Omega_D, \tag{2}$$

$$\frac{\partial\, u(x)}{\partial\, n(x)} = g_N(x) \quad \forall x \in \partial\Omega_N. \tag{3}$$

The accelerating field $\vec{E}$ can then be computed as the negative gradient $\nabla u(x)$ of the solution $u(x)$.

For ease and speed-up of development and to attain a certain flexibility in the selection of function-spaces for ansatz and test functions, we use FEniCS [13] as implemented in the C++/Python library dolfin [14] for the automated solution of the system of equations arising from the FEM formulation of Eqs. (1), (2), and (3).

FEniCS allows to directly write down the weak formulation of Poisson's equation:

$$\int_\Omega \nabla u(x) \cdot \nabla v(x)\, dx = \int_\Omega \frac{\rho(x)}{\varepsilon_0}\, v(x)\, dx \quad \forall v \in V \tag{4}$$

as a pair of a bilinear form $a(u, v)$:

$$a(u, v) = \int_\Omega \nabla u(x) \cdot \nabla v(x)\, dx \tag{5}$$

and a linear form $L(v)$:

$$L(v) = \int_\Omega \frac{\rho(x)}{\varepsilon_0}\, v(x)\, dx. \tag{6}$$

[1] This would lead to a quadratic (or better) convergence in the force with FEM, compared to a linear convergence using FD.

To be able to do this one has to import the Python module dolfin:

```
from dolfin import *
```

and to specify the discrete function spaces (depending on the mesh used):

```
V = FunctionSpace(mesh,"CG",degree)
u = TrialFunction(V)
v = TestFunction(V)
```

One can directly write down the bilinear form a as:

```
a = dot(grad(u), grad(v))*dx(mesh)
```

We can now prepare and assemble the system matrix A for the solver included in FEniCS:

```
template = PETScMatrix()
A = assemble(a,tensor=template)
```

The linear form L in the weak formulation of Eq. (1) as given in Eq. (4) depends on the charge density arising from the charges in the domain.

Starting from a constant $\rho = 0$ one can assemble the right hand side linear form L and the corresponding vector rhs:

```
L = Constant(0.0)*v*dx(mesh)
rhs = assemble(L)
```

using the charge weighting implemented by the method PointSource from dolfin to add macro-particles to the right hand side:

```
macro_particles = []
for i in range(Number_of_Particles):
  macro_particles.append((Particle[i],
      charge[i]/eps0))
delta = PointSource(V, macro_particles)
delta.apply(rhs)
```

After defining and applying the boundary condition g_D on $\partial\Omega_D$[2]

```
bc = DirichletBC(V, g_D,"on_boundary")
bc.apply(A)
bc.apply(rhs)
```

and setting up one of the solvers provided through dolfin (here the conjugate gradient method):

```
solver = PETScKrylovSolver("cg","default")
solver.parameters["relative_tolerance"] =
    residual
solver.set_operator(A)
```

one can solve for the unknown potential $u(x)$:

[2] For ease of exposition, we choose to only show the implementation using a Dirichlet boundary condition g_D on $\partial\Omega$.

ISBN: 978-3-95450-200-4 doi:10.18429/JACoW-ICAP2018-TUPAG02

```
u_x = Function(V)
solver.solve(u_x.vector(), rhs)
```

The electric field $\vec{E}$ can then be computed from the gradient $\nabla u(x)$ of the solution.

```
e_temp = -grad(u_x)
```

To be applicable as an interpolated field it has to be projected onto an appropriate function space. Possibilities include

- the matching Raviart-Thomas finite element space

```
if V_field == "RT":
  Efield = project(e_temp,
       FunctionSpace(mesh,\
  "RT", degree))
```

- the Brezzi-Douglas-Marini finite element space

```
elif V_field == "BDM":
  Efield = project(e_temp,
       FunctionSpace(mesh,\
  "BDM", degree-1))
```

- a corresponding Discontinous-Galerkin vector function space

```
elif V_field == "DG":
  Efield = project(e_temp,
       VectorFunctionSpace(mesh,\
  "DG", degree-1))
```

- and the continuous Lagrange(Courant) vector function space

```
else:
  Efield = project(e_temp,
       VectorFunctionSpace(mesh,\
  "CG", degree-1))
```

The computed field can next be used to accelerate the particles using the well-known Boris pusher [15].

FIRST RESULTS

In this section, we show first results for a simple model problem as obtained from our FEniCS implementation for MOEVE and compare with ASTRA [7]. The model problem regards tracking of an electron bunch of Gaussian distribution in all directions for a short drift space without external electromagnetic field and without ion cloud.

We have tracked an electron bunch with the new FEM solver for a drift distance of 3.0 m without any external electromagnetic field or ion cloud. The initial bunch is generated by ASTRA. The bunch profile is listed in Table 1. The rms bunch size and the emittance growth are plotted in Figures 1 and 2, respectively. The results are compared with ASTRA for the transverse directions. The emittance was computed according [16].

Table 1: Bunch Profile for Tracking

Parameters of the electron bunch	
Number of macro particles	5,000
Beam energy	15 MeV
Beam energy spread	1.49 keV
Beam charge	-0.4 nC
Transverse emittance	1.0 π mm·mrad
Bunch length	0.88 mm
rms bunch radius	0.362 mm

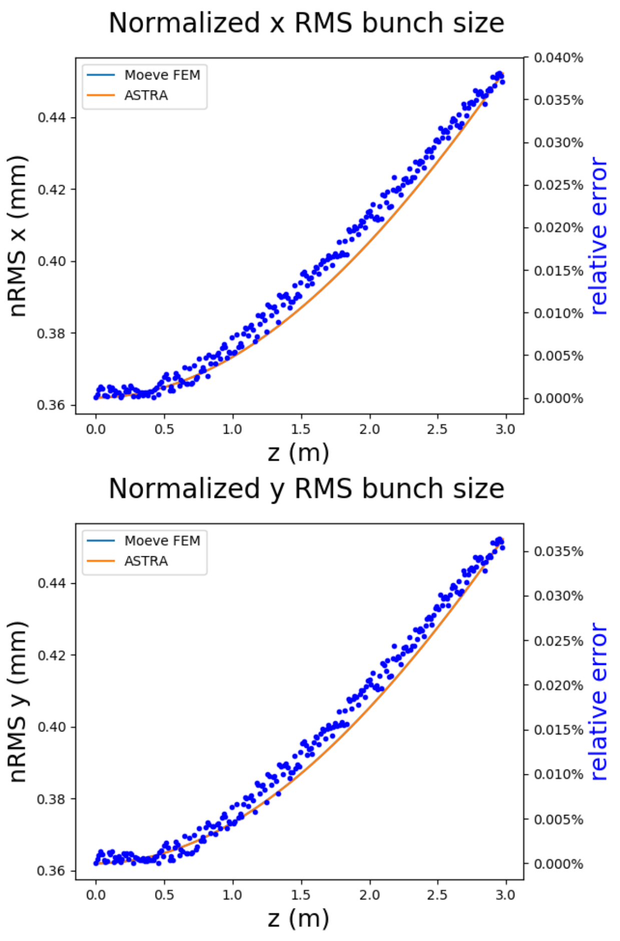

Figure 1: Bunch size growth in transverse directions for a drift distance of 3.0 m without external electromagnetic fields as computed by MOEVE based on FEM and ASTRA, respectively. Both curves agree very well. The relative error between both results is shown as well in the same plot, each.

It can be seen that the results of MOEVE with the FEM-based FEniCS implementation and ASTRA agree very well both for the transverse bunch size growth and the emittance. Regarding the relative error in the transverse bunch size growth, it is observable that the relative error grows follow-

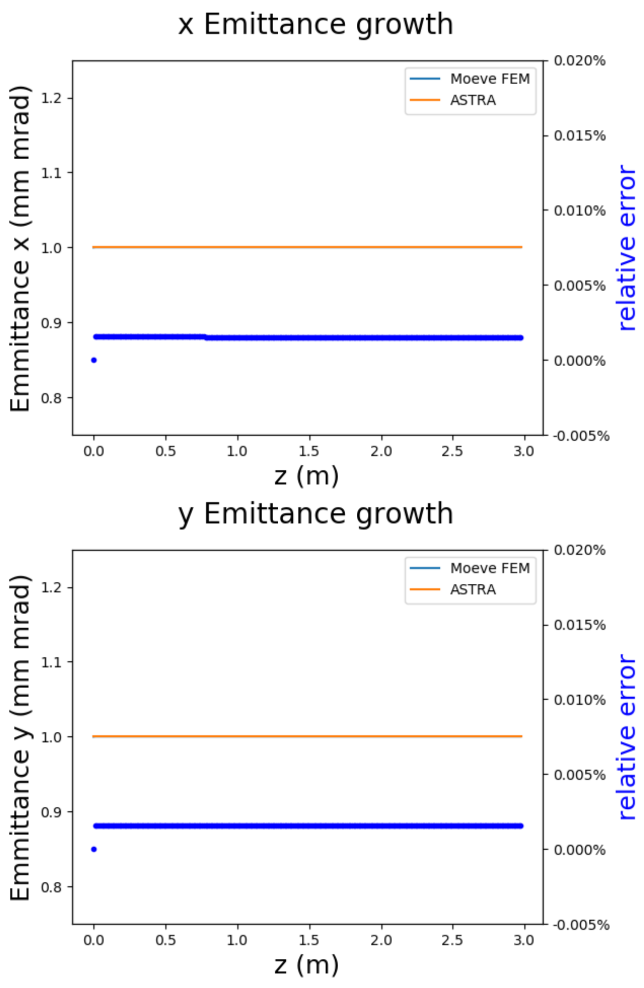

Figure 2: Emittance growth in transverse directions for a drift distance of 3.0 m without external electromagnetic fields as computed by MOEVE based on FEM and ASTRA, respectively. Both curves agree very well. The relative error between both results is shown as well in the same plot, each.

ing a very similar functional behaviour as the transverse bunch size growth itself. In that, the relative error after a drift of 3.0 m without external electric field reaches about 0.0371% and 0.0353% for x and y, respectively. On the other hand, the emittance stays constant over the drift. So does the relative error between the results of MOEVE with the FEM-based FEniCS implementation and ASTRA. Its value is about 0.0295% and 0.0317% for x and y, respectively. It should be noted that the new implementation in FEniCS needs only 8.0×10^4 DoFs for a mesh of 0.3 m in z-direction then moving along the drift distance of a 3.0 m long vacuum chamber while ASTRA uses a grid of $64^3 \approx 2.6 \times 10^5$ DoFs, which covers the bunch area only. We used an Intel Xeon workstation with 3.7 GHz CPU for both simulations. The CPU time of our current FEniCS implementation is not yet compatible to ASTRA since the procedures are not optimized yet — e.g. adaptive time stepping and other efficiency measures (see below) have not been implemented by now.

CONCLUSION AND FUTURE PERSPECTIVE

In a pilot study, we developed an FEM-based model to track electron bunches. A first study on a simple model problem, tracking through a drift space without external electric field, showed very good agreement with results obtained by ASTRA.

Next, we aim to improve this FEM-based numerical model to study ion cloud dynamics using realistic geometries for the accelerator components. To achieve a high computational performance, we will employ MPI parallelization throughout the code. Also, the charge weighting with `PointSource` as well as the speed of interpolating the accelerating field at the particles' position can still be improved.

Additionally, we want to use the flexibility of the FEM approach to implement adaptive hp-refinements, i.e. with respect to elemnt size (h) and to the polynomial degree (p) of the FEM approach. Elements that have macro-particles allocated to them (e.g. that are close to the bunch or covering it) should use a low-order approximation and be small [17], while further away from high charge densities one can use larger elements with a high order of approximation.

For validation, we will employ measurement results obtained at the Electron Stretcher Accelerator (ELSA) in Bonn [18]. Then, we will study the ion cloud dynamics to be expected in the proposed ERL bERLinPro at Helmholtz-Zentrum Berline (HZB) [10]. In general, this numerical model can serve for ion cloud studies to estimate and reduce the effects related to ionized residual gas in high-current electron machines.

REFERENCES

[1] G. Pöplau. MOEVE: Multigrid Poisson Solver for Non-Equidistant Tensor Product Meshes. Germany, (2003).

[2] G. Pöplau, A. Meseck, and U. van Rienen. Simulation of the interaction of an electron beam with ionized residual gas, in *Proc. IPAC 2011*, 2011, pp. 2250-2252.

[3] G. Pöplau, U. van Rienen, A. Meseck. Numerical studies of the behavior of ionized residual gas in an energy recovering linac, *Phys. Rev. ST Accel. Beams 18*, 044401 (2015).

[4] G. Pöplau, et al. Simulations for ion clearing in an ERL, in *Proc. ICAP 2012*, 2012.

[5] A. Markovic , G. Pöplau, and U. van Rienen. Computation of the 2D transverse wake function of an electron-cloud for different parameters, in *Proc. IPAC 2012*, 2012.

[6] S. van der Geer, M. de Loos. The general particle tracer code. Design implementation and application, 2001.

[7] K. Flöttmann. ASTRA: A space charge tracking algorithm. Manual, Version 3 (2011): 2014.

[8] G. H.Hoffstaetter, M. Liepe. Ion clearing in an ERL, *Nuclear Instruments and Methods in Physics Research Section A 557*, 2006, pp. 205-212

[9] E. Brentegani, W. Hillert, A. Meseck, D. Sauerland, U. van Rienen. Simulation of the Distribution of Parasitic Ions in the Potential of an Electron Beam, in *Proc. ICAP 2015*, Shanghai, China (2015).

ISBN: 978-3-95450-200-4 doi:10.18429/JACoW-ICAP2018-TUPAG02

[10] B. Kuske, N. Paulick, A. Jankowiak, J. Knobloch. Conceptual Design Report (CDR). HZB, Berlin, Germany (2011).

[11] A. Markovic. Simulation of the interaction of positively charged Beams and electron clouds. PhD Thesis. Rostock University, 2013.

[12] C. Bahls. Space charged calculations using refinements on structured and unstructured grids. PhD Thesis. Rostock University, 2015.

[13] A. Logg, K.-A. Mardal, G. N. Wells et al. Automated Solution of Differential Equations by the Finite Element Method. Springer, 2012

[14] A. Logg, G. N. Wells and J. Hake. DOLFIN: a C++/Python Finite Element Library. Automated Solution of Differential Equations by the Finite Element Method. Volume 84 of Lecture Notes in Computational Science and Engineering, Edited by A. Logg, K.-A. Mardal and G. N. Wells, Springer, chapter 10, 2012

[15] J.B. Boris. Relativistic plasma simulation-optimization of a hybrid code", in *Proceedings of the 4th Conference on Numerical Simulation of Plasmas*, November 1970. Naval Res. Lab., Washington, D.C., pp. 3-67.

[16] K. Floettmann. Some basic features of the beam emittance, *Phys. Rev. ST Accel. Beams 6*, 034202, 2005.

[17] C. Bahls, G. Poeplau, U. van Rienen. Using Nudg++ to Solve Poisson's Equation on Unstructured Grids, in *Proc. Scientific Computing in Electrical Engineering SCEE 2008*. Springer, 2010.

[18] D. Sauerland, W. Hillert, A. Meseck. Estimation of the ion density in accelerators using the beam transfer function technique, in *Proc. IPAC 2015*, Richmond, VA, USA (2015).